НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

«На правах рукопису»
УДК ____519.21_____

«До захисту допущено»
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
“15” травня 2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності_____113 «Прикладна математика»_____
(код і назва)

на тему: _____«Порівняльний аналіз сучасних схем гешування паролів»_____

Виконала: студентка __2_ курсу, групи __ФІ-63м__
(шифр групи)

_____Бурлака Валерія Володимирівна_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____доцент кафедри ММЗІ, к.т.н. Яковлєв С.В._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Фізико-технічний інститут
Кафедра математичних методів захисту інформації

Рівень вищої освіти – другий (магістерський)

Спеціальність 113 «Прикладна математика»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Бурлаці Валерії Володимирівні
(прізвище, ім'я, по батькові)

1. Тема дисертації Порівняльний аналіз сучасних схем гешування паролів
_____,
науковий керівник дисертації Яковлєв Сергій Володимирович, к.т.н.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 02.07.2018р. № 1058-с

2. Термін подання студентом дисертації 16.05.2018р.

3. Об'єкт дослідження Криптографічні процеси у системах захисту інформації

4. Предмет дослідження Схеми гешування паролів та методи їх аналізу
_____
_____

5. Перелік завдань, які потрібно розробити Провести огляд опублікованих джерел за темою дослідження; ознайомитися з криптографічними властивостями, які повинна мати схема гешування паролів з огляду на розвиток сучасних методів криптоаналізу; розглянути конструкцію схем-фіналістів конкурсу «Password Hashing Competition»; сформулювати критерії для порівняння схем гешування паролів, порівняти схеми за цими критеріями та визначити схеми, придатні для впровадження геш-функції

«Купина»; сформулювати алгоритм роботи схеми гешування паролів із зазначеною геш-функцією та провести первинну оцінку надійності одержаних схем.

6. Орієнтовний перелік ілюстративного матеріалу 6 рисунків, 3 таблиці

7. Орієнтовний перелік публікацій

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Узгодження теми з науковим керівником	20.09.17 – 04.10.17	
2.	Пошук літератури та опрацювання знайденого матеріалу	05.10.17 – 06.11.17	
3.	Ознайомлення з властивостями схем гешування паролів	07.11.17 – 27.11.17	
4.	Огляд конструкцій схем-фіналістів конкурсу PHC	28.11.17 – 25.12.17	
5.	Визначення критеріїв для порівняння схем	09.01.18 – 31.01.18	
6.	Порівняння схем за критеріями та визначення схем, придатних для впровадження геш-функції «Купина»	01.02.18 – 12.03.18	
7.	Опис алгоритму роботи нових схем гешування паролів та проведення первинної оцінки надійності одержаних схем	13.03.18 – 13.04.18	
8.	Оформлення роботи	14.04.18 – 14.05.18	

Студент _____ В.В. Бурлака
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ С.В. Яковлєв
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника магістерської дисертації.

РЕФЕРАТ

Кваліфікаційна робота містить: 52 стор., 6 рисунків, 3 таблиці, 30 джерел.

Наразі для надійного зберігання паролів проводиться їх обробка за спеціальними схемами. У роботі було розглянуто вимоги до захисту схем гешування паролів. Також було зосереджено увагу на схемах гешування паролів, що стали фіналістами конкурсу «Password Hashing Competition», проведено їх порівняльний аналіз за певними критеріями та визначено схеми, придатні для впровадження в них геш-функції «Купина».

Метою роботи є порівняльний аналіз сучасних схем гешування паролів та їх адаптація під національні криптографічні стандарти, що дозволить використовувати сучасні схеми парольного гешування у засобах криптографічного захисту, призначених для державних органів України.

Об'єктом дослідження є криптографічні процеси у системах захисту інформації.

Предметом дослідження є схеми гешування паролів та методи їх аналізу.

У результаті роботи було запропоновано схеми гешування паролів, що використовують у якості криптографічного примітиву геш-функцію «Купина», яка є національним стандартом криптографічного захисту інформації в Україні.

ПАРОЛЬ, ГЕШУВАННЯ ПАРОЛІВ, СХЕМА ГЕШУВАННЯ ПАРОЛІВ, ГЕШ-ФУНКЦІЯ «КУПИНА», ДСТУ 7564:2014, PASSWORD HASHING COMPETITION, ARGON2, CATENA

ABSTRACT

Thesis consists of 52 pages, 6 illustrations, 3 tables, 30 literature sources.

At present, passwords are processed by special schemes for secure storage. The requirements for protecting of password hashing schemes were considered in this paper. Also, attention was focused on password hashing schemes that became the finalists of the Password Hashing Competition, their comparative analysis according to certain criteria was carried out and the schemes suitable for implementation in them of the Kupyna hash function were defined.

The aim of the work is comparative analysis of modern password hashing schemes and their adaptation to the national cryptographic standards, which will allow using of modern password hashing schemes in the means of cryptographic protection that intended for state bodies of Ukraine.

The object of the study is the cryptographic processes in the systems of information protection.

The subject of the study is the password hashing schemes and methods of their analysis.

As a result of the work, there were proposed password hashing schemes that use the Kupyna hash function as a cryptographic primitive, which is the national standard of cryptographic protection of information in Ukraine.

PASSWORD, PASSWORD HASHING, PASSWORD HASHING SCHEMES, KUPYNA HASH FUNCTION, DSTU 7564: 2014, PASSWORD HASHING COMPETITION, ARGON2, CATENA

ЗМІСТ

ВСТУП

Актуальність дослідження. Найбільш розповсюдженим способом для перевірки справжності користувачів при доступі до комп'ютерних систем та Інтернет-сервісів є однофакторна автентифікація, що вимагає від користувача пред'явлення секретної фрази, тобто паролю. У зв'язку з цим постає проблема надійного зберігання паролів користувачів на сервері автентифікації задля унеможливлення отримання зловмисником доступу до чужого облікового запису.

Досягання у розробці апаратних та програмних засобів збільшили можливості здійснення атак на відомі широко застосовні схеми гешування паролів, що спричинило необхідність розробки нових. Саме через це в 2013 році міжнародною криптографічною спільнотою було оголошено конкурс «Password Hashing Competition» [1], у результаті якого було визначено нові схеми гешування паролів, що мають якісні криптографічні властивості та ефективно реалізуються на сучасних платформах. Однак наразі відсутні схеми парольного гешування, які б ґрунтувалися на національних стандартах криптографічного захисту інформації, зокрема, геш-функції «Купина», що описана у ДСТУ 7564:2014 «Інформаційні технології. Криптографічний захист інформації. Функція гешування» [2], і, відповідно, могли б використовуватися для захисту даних на національному рівні.

Метою дослідження є порівняльний аналіз сучасних схем гешування паролів та їх адаптація під національні криптографічні стандарти, що дозволить використовувати сучасні схеми парольного гешування у засобах криптографічного захисту, призначених для державних органів України. Для досягнення мети необхідно вирішити такі **завдання**:

1) провести огляд опублікованих джерел за темою дослідження;

2) ознайомитися з криптографічними властивостями, які повинна

мати схема гешування паролів з огляду на розвиток сучасних методів криптоаналізу; розглянути конструкцію схем-фіналістів конкурсу «Password Hashing Competition»;

3) сформулювати критерії для порівняння схем гешування паролів, порівняти схеми за цими критеріями та визначити схеми, придатні для впровадження геш-функції «Купина»;

4) сформулювати алгоритм роботи схем гешування паролів із зазначеною геш-функцією та провести первинну оцінку надійності одержаних схем.

Об'єктом дослідження є криптографічні процеси у системах захисту інформації.

Предметом дослідження є схеми гешування паролів та методи їх аналізу.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи системного аналізу та системного підходу, дискретної математики, математичного та комп'ютерного моделювання.

Наукова новизна отриманих результатів полягає в тому, що вперше запропоновано схеми гешування паролів, що використовують у якості криптографічного примітиву геш-функцію «Купина», що є національним стандартом криптографічного захисту інформації України.

Практичне значення результатів полягає в можливості використання модифікованих схем гешування паролів для захисту інформації на національному рівні, оскільки ці схеми включають геш-функцію, що визначена національним стандартом ДСТУ 7564:2014 «Інформаційні технології. Криптографічний захист інформації. Функція гешування».

1 СХЕМИ ГЕШУВАННЯ ПАРОЛІВ: СТАН РОЗВИТКУ

Парольна автентифікація є найбільш розповсюдженим способом перевірки справжності користувачів в комп'ютерних системах. Вона полягає у введенні користувачем секретної фрази, що відповідає його ідентифікатору. Зазначена інформація порівнюється з еталонним паролем, що зберігається у спеціальній базі даних, у результаті чого приймається рішення про особистість користувача та надання доступу до інформаційних ресурсів. Така автентифікація вирізняється своєю простотою та звичністю, проте має суттєві недоліки через уразливість методів обробки паролів для їх зберігання.

В даному розділі буде розглянуто метод гешування паролів, що використовується для посилення надійності зберігання інформації пов'язаної з користувачем, а також результати в розробці таких методів на теперішній час.

1.1 Гешування як метод захисту паролів

Пароль є пам'ятним секретом користувача, що складається з декількох друкованих символів. Одним із способів для його безпечного зберігання є підходи з використанням криптографічної геш-функції. Таким чином, після встановлення користувачем паролю для свого облікового запису, пароль обробляється геш-функцією і надалі зберігається в системі не як відкритий текст, а як геш. З кожним разом, коли користувач бажає отримати доступ до сервісу, він вводить пароль, зазначені дані гешуються та порівнюється зі збереженим значенням, у результаті чого робиться висновок про надання доступу користувачу.

Як відомо, криптографічно безпечні геш-функції практично

неможливо інвертувати, так що компрометація бази даних користувачів не призведе до безпосереднього розкриття користувацьких даних. Однак зловмисник може здійснювати відомі криптографічні атаки для знаходження паролю. Тоді він може отримати доступ до облікового запису подібно до його законного власника.

Найбільш розповсюдженим є використання користувацьких секретів довжиною вісім символів – 8 байт в кодуванні ASCII. Проте часто такі паролі мають низьку ентропію, через що вони можуть бути з легкістю вгадані зловмисником методом повного перебору. Саме використання геш-функції, що, як відомо, перетворює вхід довільної довжини на вихід фіксованої довжини, наприклад, в 32 або 64 байти, вже сповільнює атаку. Також з метою захисту від вичерпного пошуку використовується техніка key stretching [3], яка передбачає багаторазове застосування геш-функції. Так, якщо X — пароль з μ біт ентропії, а H — криптографічна геш-функція, то супротивник, який знає геш пароля $Y_1 = H(X)$, може розраховувати на пошук X, випробовуючи близько $2^{\mu-1}$ кандидатів. Щоб уповільнити зловмисника в 2^δ рази, необхідно стільки ж раз застосувати геш-функцію, тобто обчислити $Y_i = H(Y_{i-1})$ для $i \in \{2, \ \ldots, \ 2^\delta\}$, а потім використовувати $Y_{2\delta}$ як остаточний геш пароля. Це змушує супротивника викликати геш-функцію $2^{\delta+\mu}$, а не 2^μ разів, але також збільшує час обчислення гешу і для користувача. Однак в такий спосіб можна лише сповільнити роботу зловмисника, а не зовсім унеможливити атаку вичерпного пошуку.

Також зловмисник може спробувати здійснити атаку компромісу «час-пам'ять» [4], наприклад, з використанням райдужних таблиць [5]. У такому випадку він попередньо обчислює можливі значення геш-функції та зберігає результати у таблиці пошуку. Таким чином, основна частина роботи виконується лише один раз під час створення таблиці. Кожна наступна атака вимагає лише пошуку по таблиці, а не ітеративного обчислення геш-функції.

Окрім можливість здійснення описаних атак, постає також інша

проблема: якщо декілька користувачів встановлюють однаковий пароль, результати гешування також будуть однаковими, а тому при визначенні зловмисником паролю до одного облікового запису, вразливими стають облікові дані всіх користувачів. Так само, якщо один користувач використовує один і той самий секрет для доступу до багатьох облікових записів, розкриття одного з них викличе стурбованість щодо безпека інших. Для запобігання цього використовується параметр сіль — рядок даних, який зазвичай складається з 8 випадкових байтів. Під час створення нового облікового запису генерується випадкова сіль, яка разом з парольною фразою подається на вхід функції гешування. Таким чином, однаковий секрет має різні геші для різних облікових записів, що запобігає збігу гешів, створених за тією самою секретною фразою.

Сіль зберігається разом з гешем таємниці як відкритий текст. Операція автентифікації аналізує сіль та пароль процедури входу та порівнює результат із збереженим гешовим значенням пароля для перевірки справжності користувача. На рисунку 1.1 наведена графічно розглянута схема обробки паролів.

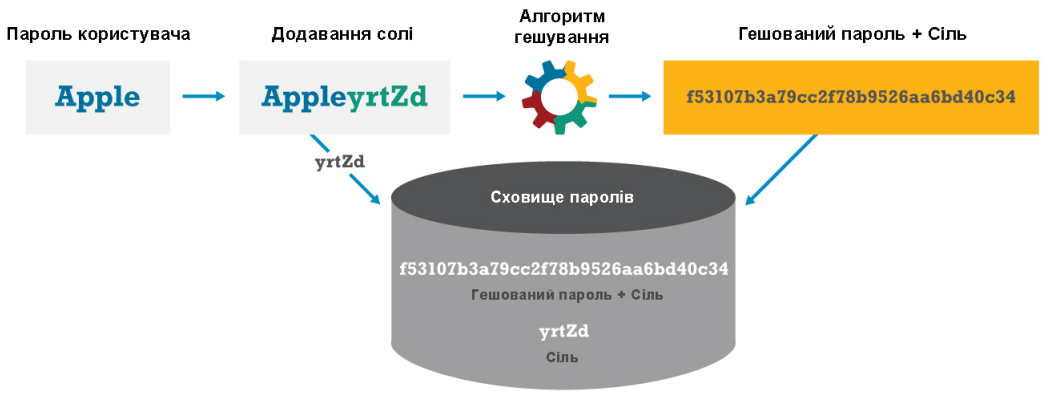

Рисунок 1.1 – Схема обробки паролів з використання солі [6]

Таким чином, використання солі додає певний ступінь унікальності кожному окремому гешу пароля, запобігає відповідності ідентичним паролям однакових гешів, уповільнює виконання атак повного перебору, а

також унеможливлює атаки за попередньо обчисленою таблицею пошуку, оскільки її наявність робить попередні обчислення просто позбавленими сенсу.

Через відсутність повністю захищеної геш-функції, обробка паролю для його подальшого зберігання лише геш-функцією не є достатньою. Тому були запропоновані спеціальні схеми гешування паролів, такі як PBKDF2 [7], Bcrypt [8] та Scrypt [9].

Однак прогрес в розробці спеціалізованих апаратних засобів та паралельних обчисленнях дозволив здійснювати атаки, які долають захист цих схем. Нападники, що використовують графічні процесори (ГП), програмовані логічні інтегральні схеми (ПЛІС) та інтегральні схеми спеціального призначення (ІССП), мають значні можливості для розкриття інформації користувача, випробовуючи багато кандидатів в паролі паралельно. Схеми PBKDF2 та Bcrypt є вразливими до таких атак. Задля протидії їм було розроблено обчислювально складні за пам'яттю схеми гешування паролів. Такою є схема Scrypt, але, на жаль, вона уразлива до інших відомих атак, пов'язаних зі спостереженням за пам'яттю [10].

1.2 Вимоги безпеки до схем гешування паролів

Технологічні досягнення та широка доступність потужних апаратних засобів разом з програмним забезпеченням для зламу роблять можливим розкриття гешів паролів з параметром сіль і без нього. Рішення цієї проблеми зводиться до уповільнення зловмисника при здійсненні розрахунків. Загальноприйнятою технікою для досягнення цього є багаторазове застосування геш-функції до паролю. Однак використання спеціалізованих апаратних засобів, наприклад, ГП, ПЛІС, ІССП, тощо, робить можливим злам обмежених швидкістю роботи центрального процесора (ЦП) гешів для добре укомплектованого

зловмисника. Для вирішення цієї проблеми були введені геш-функції з обмеженням за пам'яттю, тому будь-яка сучасна схема гешування паролів має бути обмежена за цими двома параметрами. Також існують й інші вимоги до схем, які повинні бути враховані на етапі розробки.

Наведемо список основних бажаних властивостей захисту схем гешування паролів, ґрунтуючись на типах атак, з якими вони можуть стикнутися [11]:

– *Криптографічна безпека.* Схема повинна бути криптографічно захищеною та повинна володіти наступними властивостями:

1) стійкість до пошуку прообразу;

2) стійкість до пошуку другого прообразу;

3) стійкість до колізій.

Крім того, слід уникати інших криптографічних уразливостей таких, як ті, що містяться в деяких конструкціях Меркле-Дамгара, наприклад, уразливість до атак розширення довжини, часткові колізії повідомлень, тощо.

– *Захист від таблиць пошуку / атак компромісу «час-пам'ять».* Схема повинна бути сконструйована так, щоб атаки компромісу «час-пам'ять», які дозволяють створювати попередньо обчислені таблиці пошуку, такі як райдужні таблиці, були неможливі.

– *Захист від ЦП-оптимізованих зловмисників.* Схема повинна бути ЦП-важкою, тобто повинна вимагати значних обсягів обробки процесором так, щоб її роботу не можна було оптимізувати за допомогою програмних чи апаратних засобів. Тому такі оптимізовані для злому (багатоядерні) процесори повинні пропонувати лише мінімальні покращення швидкості обчислень ніж ті, що призначені для захисту паролю.

– *Захист від апаратно-оптимізованих зловмисників.* Схема повинна бути обчислювально складною за пам'яттю, тобто для обчислення вона повинна вимагати значних обсягів оперативної пам'яті так, щоб її роботу не можна було оптимізувати, наприклад,

використовуючи атаки компромісу «час-пам'ять». Такі оптимізовані для зламу реалізації на ІССП, ПЛІС та ГП повинні забезпечувати лише мінімальні удосконалення ніж ті, що призначені для захисту пароля.

– *Захист від атак за побічними каналами.* При розробці схеми гешування паролів потрібно зосередити увагу на безпеці від такого типу атак за побічними каналами, як cache-timing атаки [12] з урахуванням існування таких атак на відому схему Scrypt [9]. Також необхідно забезпечити захист від Garbage-Collector атак (GCA) [13]. GCA в застосуванні до схем гешування паролів є прикладом атаки «витоку пам'яті». Ці атаки складаються зі сценарію, де зловмисник має доступ до внутрішньої пам'яті машини жертви під час або після закінчення роботи схеми гешування, або в якийсь момент, коли сам пароль залишається в пам'яті. Обидва типи атак використовують спостереження за пам'яттю, щоб знайти деяке (проміжне або залишкове) значення y, отримане як результат функції F при поданні на вхід пароля p, де для тестування кандидатів пароля p', використовуючи F, потрібно докласти значно менше зусиль порівняно з оригінальною схемою гешування, що зменшує загальну безпеку.

Таким чином, при розробці безпечної схеми гешування паролів необхідно обов'язково врахувати перераховані вище вимоги.

Надалі в роботі зосереджується увага на усіх з описаних вище пунктах, окрім пункту про захист від ЦП-оптимізованих зловмисників, оскільки аналіз цієї властивості вимагає програмної реалізації на спеціалізованих апаратних засобах, що тягне за собою великі матеріальні витрати, яких, на жаль, автор роботи не має.

Далі розглянемо детально атаки компромісу «час-пам'ять», оскільки вони прості в здійсненні при зберіганні пароля у вигляді гешу, а також дозволяють зловмиснику визначити секрет користувача з мінімальними витратами часу, але з використанням великих обсягів пам'яті.

1.3 Огляд атак компромісу «час-пам'ять»

Криптоаналітичні атаки, що базуються на вичерпному пошуку, потребують великої обчислювальної потужності або багато часу для здійснення. Коли така ж атака повинна виконуватися декілька разів, можливо, можна буде виконати вичерпний пошук заздалегідь і зберегти всі результати в пам'яті. Після того, як ці попередні обчислення буде зроблено, атака може бути здійснена практично миттєво. На жаль, цей метод недоцільний через великий обсяг необхідної пам'яті.

У 1980 році Мартін Геллман описав криптоаналітичний компроміс «час-пам'ять» [4], що дозволяє обмінювати кількість використовуваної пам'яті на час здійснення атаки. Так, для криптосистеми, що має N ключів, цей метод може відновити ключ за $N^{2/3}$ операцій, використовуючи $N^{2/3}$ слів пам'яті. Типовим застосуванням методу є відновлення ключа при відомих відкритому та шифрованому текстах. Однак метод може бути застосований й у галузі гешування паролів. Багато популярних операційних систем генерують геші паролів шляхом шифрування фіксованого відкритого тексту з паролем користувача в якості ключа і зберігають результату як геш пароля. Якщо схема гешування паролів погано розроблена, відкритий текст та метод шифрування будуть однаковими для всіх паролів. У цьому випадку геші паролів можна обчислити заздалегідь, що відповідає компромісу «час-пам'ять».

Варто зазначити, що компроміс «час-пам'ять» є імовірнісним методом. Успіх не гарантується, а коефіцієнт успішності залежить від часу та пам'яті, що виділяються для криптоаналізу.

Розглянемо детальніше метод, що був запропонований Геллманом.

Нехай P_0 — фіксований відкритий текст C_0 – відповідний шифрований текст, метод намагається знайти ключ $k \in N$, який використовувався для шифрування відкритого тексту за допомогою

шифру S. Отже, маємо:

$$C_0 = S_k(P_0).$$

Надалі заздалегідь генеруються всі можливі шифротексти шляхом шифрування відкритого тексту на всіх N ключах. Шифротексти утворюють ланцюги, причому в пам'яті зберігаються лише перший і останній елементи ланцюга. Збереження лише першого і останнього елементу ланцюга — операція, яка реалізує компроміс — економію пам'яті за рахунок часу криптоаналізу. Ланцюги утворюються за допомогою функції редукції R, яка створює ключ із шифротексту, оскільки шифротекст довший за ключ. Послідовно застосовуючи шифр S та функцію редукції R, створюються ланцюги змінних ключів і шифрованих текстів:

$$k_i \xrightarrow{S_{k_i}(P_0)} C_i \xrightarrow{R(C_i)} k_{i+1}.$$

Послідовність $R(S_k(P_0))$, позначена через $f(k)$, генерує ключ із ключа, що призводить до формування ланцюгів ключів:

$$k_i \xrightarrow{f} k_{i+1} \xrightarrow{f} k_{i+2} \to \dots.$$

Таким чином, створено m ланцюгів довжини t, і їх перший і останній елементи зберігаються в таблиці. Для певного шифротексту C можна спробувати з'ясувати, чи використовується ключ для генерації C, серед тих, що використовуються для створення таблиці. Для цього генерується ланцюг ключів, починаючи з $R(C)$, довжини t. Якщо C дійсно був отриманий за допомогою ключа, який використовувався при створенні таблиці, згодом буде отримано ключ, який відповідає останньому ключовому елементу відповідного ланцюга. Цей останній ключ був збережений у пам'яті разом з першим ключем ланцюга. Використовуючи перший ключ ланцюга, можна відновити весь ланцюг, і зокрема ключ, який йде безпосередньо перед $R(C)$. Цей ключ є ключем, який використовувався для генерації C, і є шуканим ключем.

На жаль, є шанс, що ланцюги, які починаються з різних ключів,

стикаються та зливаються. Це пов'язано з тим, що функція R довільним чином переводить простір шифротекстів у простір ключів. Чим більша таблиця, тим вища ймовірність того, що новий ланцюг зіллється з одним із попередніх. Кожне злиття зменшує кількість окремих ключів, які насправді охоплені таблицею. Ймовірність знайти ключ, використовуючи таблицю з m рядків та t ключів, є наступною:

$$P_{table} \geq \frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} \left(1 - \frac{it}{N}\right)^{j+1}.$$

Ефективність однієї таблиці зменшується зі збільшенням її розміру. Щоб отримати високу ймовірність успіху, краще створити кілька таблиць, використовуючи різну функцію редукції для кожної таблиці. Ланцюжки різних таблиць можуть зіткнутися, але не зливатимуться, оскільки в різних таблицях використовуються різні функції редукції. Ймовірність успіху при використанні l таблиць визначається формулою:

$$P_{success} \geq 1 - \left(1 - \frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} \left(1 - \frac{it}{N}\right)^{j+1}\right)^{l}.$$

Під час пошуку ключа в таблиці, пошук відповідної кінцевої точки не означає, що ключ знаходиться в таблиці. Дійсно, ключ може бути частиною ланцюга, який має одну кінцеву точку, але не знаходиться в таблиці. У цьому випадку генерація ланцюга зі збереженої відправної точки не дає ключ, що називається помилковою тривогою. Помилки також виникають, коли ключ знаходиться в ланцюгу, який є частиною таблиці, але зливається з іншими ланцюгами таблиці. У цьому випадку декілька вихідних точок відповідають одній і тій же кінцевій точці, і, можливо, доведеться створювати декілька ланцюгів, допоки ключ не буде остаточно знайдено.

До 1982 року Райвестом було запропоновано [14] використовувати виділені точки як кінцеві точки для ланцюгів. Виділені точки — це точки, для яких справедливий простий критерій, наприклад, перші десять бітів

ключа дорівнюють нулю. Усі кінцеві точки, що зберігаються в пам'яті, є виділеними. При отриманні першого шифротексту, створюється ланцюжок ключів допоки не знайдеться виділена точка, і тільки тоді буде виконано пошук її в пам'ять. Такий спосіб дії значно зменшує кількість пошуків в пам'яті.

Існує три параметри, які можна зкорегувати у компромісі «час-пам'ять»: довжина ланцюгів t, кількість ланцюгів на таблицю m та кількість створених таблиць l.

Ці параметри можна регулювати, щоб задовольнити межам пам'яті M, часу криптоаналізу T і успіху $P_{success}$. Межа ймовірності успіху подана рівнянням вище. Обмеження на пам'ять визначається кількістю ланцюгів на таблицю m, кількістю таблиць l і кількістю пам'яті m_0, необхідною для зберігання вихідної та кінцевої точки. Обмеження на час T визначається середньою довжиною ланцюгів t, кількістю таблиць l і швидкістю $\frac{1}{t_0}$, за якої відкритий текст може бути зашифрований. Ці обмеження пов'язані з найгіршим випадком, коли потрібно виконувати пошук за всіма таблицями, але в даному випадку не враховується час, витрачений на помилкові тривоги. Таким чином, маємо:

$$M = m \times l \times m_0 \qquad T = t \times l \times t_0.$$

Основним обмеженням описаної схеми є той факт, що коли дві ланцюги стикаються в одній таблиці, вони зливаються. Тому Філіпом Йошліном було запропоновано [5] новий тип ланцюгів, які можуть зіткнутися в межах однієї таблиці без об'єднання. Ці ланцюги отримали назву райдужні. Вони використовують послідовну функцію редукції для кожної точки ланцюга. Вони починаються з функції редукції 1 і закінчуються функцією редукції $t - 1$. Таким чином, якщо два ланцюги стикаються, вони зливаються тільки тоді, коли зіткнення виявляється в одній і тій же позиції в обох ланцюгах. Якщо зіткнення не з'являється в тій же позиції, обидва ланцюги будуть продовжуватись, використовуючи

іншу функцію редукції, і, таким чином, не зіллються у майбутньому. Якщо відбувається зіткнення ланцюгів довжини t, то ймовірність їх злиття, таким чином, становить лише $\frac{1}{t}$. Ймовірність успіху в межах одної таблиці розміру $m \times t$ визначається як:

$$P_{table} = 1 - \prod_{i-1}^{t} \left(1 - \frac{m_i}{N}\right),$$

де $m_1 = m$ та $m_{n+1} = N\left(1 - e^{-\frac{m_n}{N}}\right)$.

Знаходження ключа по райдужній таблиці відбувається наступним чином. Спочатку застосовується R_{n-1} до шифротексту та шукається результат серед кінцевих точок таблиці. Якщо кінцеву точку буде знайдено, ланцюжок перебудовується, використовуючи відповідну відправну точку. Якщо кінцеву точку не знайдемо, необхідно знайти її, застосовуючи R_{n-2}, f_{n-1}, щоб побачити, чи знаходиться ключ у другому останньому стовпці таблиці. Потім застосовуються R_{n-3}, f_{n-2}, f_{n-1} і так далі. Тож загальна кількість розрахунків, яку необхідно зробити, складає $\frac{t(t-1)}{2}$. Це вдвічі менше, ніж у класичному методі. Дійсно, потрібні провести t^2 обчислень для пошуку відповідних t таблиць розміру $m \times t$ [5].

Таким чином, атаки компромісу «час-пам'ять» дозволяють зловмиснику виконати пошук користувацької парольної фрази, що зберігається у вигляді гешу, за малу кількість часу, але при цьому використовуючи великий обсяг пам'яті. Однак застосування атаки стає неможливим, якщо при гешуванні до паролю додається параметр сіль, оскільки така ситуація вимагає великого обсягу попередніх обчислень і, відповідно, пам'яті для їх зберігання, що робить застосування даного методу криптоаналізу недоцільним.

Далі розглянемо обчислювально складні за за пам'яттю функції, які своїм розвитком зробили вагомий внесок у концепцію розробки схем гешування паролів.

1.4 Обчислювально складні за пам'яттю функції

Ідеальна функція гешування паролів повинна висувати зловмиснику ті ж самі вимоги для свого обчислення, що й необхідні для законного підрахунку функції на сервері автентифікації. Використання тільки криптографічної геш-функції для обробки паролів зазнає невдачі в цьому відношенні: обчислення гешу на x86-центральному процесорі загального призначення, який буде використовуватися сервером автентифікації, у більшості випадків вимагає набагато більше енергії, ніж обчислення геш-функції на обладнанні спеціального призначення, наприклад, ІССП, яке використовуватиме зловмисник. У зв'язку з цим з'явилися схеми, такі як PBKDF2 [7] та Bcrypt [8], де певний блок обчислень повторюється багато разів, щоб хоч таким способом уповільнити розрахунки зловмисника. Однак з часом нападники іммігрували на нові архітектури, що дозволили швидко виконувати обчислення, які майже не вимагають пам'яті, але вони відчули труднощі при роботі з великою кількістю пам'яті. Так з'явилася необхідність в розробці обчислювально складних за пам'яттю функцій. Для свого обчислення вони потребують великої кількості пам'яті і накладають обчислювальні штрафи, якщо використовується менше пам'яті. Такі функції покликані закрити розрив продуктивності між атакуючим і захисником.

Дизайн обчислювально складних за пам'яттю функції виявився серйозною проблемою. З початку 80-х років було відомо, що багато криптографічних проблем, які, здавалося б, потребують великих обсягів пам'яті, фактично дозволяють компроміс «час-пам'ять» [4], де противник може обмінювати пам'ять на час і виконувати свою роботу на швидкому обладнанні з меншою кількістю пам'яті. У застосуванні до схем гешування паролів це означає, що злам паролів все ще може бути реалізований на спеціальному обладнанні навіть за певних додаткових витрат.

Обчислювально складні за пам'яттю функції стали важливим інструментом при розробці схем гешування паролів, оскільки їх неможливо ефективно обчислювати використовуючи спеціалізовані апаратні засоби. Вперше такі функції були описані Коліном Персивалем [9] у вигляді алгоритму Scrypt. З тих пір вони отримали зростаюче схвалення в криптографічній спільноті. Їх було запропоновано для гешування паролів для використання у зберіганні облікових даних на сервері входу та як функції виведення ключа у криптографії на основі пароля, і наразі вони все частіше використовуються з цими цілями.

Персивалем було дано наступне визначення обчислювально складної за пам'яттю функції.

Нехай g позначає коефіцієнт вартості пам'яті. Обчислювально складна за пам'яттю функція f — це функція, що може бути обчислена на машині з довільним доступом до пам'яті, використовуючи простір $S(g)$ і $T(g)$ операцій, де $S(g) \in \Omega(T(g)^{1-\alpha})$ для всіх $\alpha > 0$.

Таким чином, обчислювально складний за пам'яттю алгоритм — це алгоритм, який асимптотично використовує практично стільки пам'яті, скільки він використовує операції. Також його можна розглядати як алгоритм, який наближається до використання максимально можливого обсягу пам'яті для певної кількості операцій.

Вимога щодо обсягу пам'яті приблизно пропорційного кількості виконуваних операцій необхідна для досягнення мети створення дорогих апаратних реалізацій, залишаючись в межах ресурсів, доступних для впровадження програмного забезпечення [9].

Однак при використанні таких функції для гешування паролів вони повинні вимагати використання фіксованої кількості пам'яті під час проведення розрахунків. При використанні меншої, ніж цієї заздалегідь заданої та фіксованої кількості пам'яті, підрахунок повинен займати значно більше часу, бажано, щоб він зростав експоненційно [15].

Обчислювально складні за пам'яттю функції загалом можна розділити на дві категорії в залежності від способу доступу до пам'яті:

залежний та незалежний від даних.

Під залежним від даних доступом до пам'яті розуміють залежність від таких вхідних даних схеми гешування паролів, як пароль та сіль. Такий підхід не дозволяє зловмиснику вибрати та попередньо обчислити відсутні дані. Супротивник визначає, що він повинен перерахувати, лише тоді, коли елемент потрібен. Однак такі схеми вразливі до атак за побічними каналами, і, відповідно, тимчасова інформація дозволяє значно швидше знайти пароль.

При незалежному від даних доступі адреси пам'яті можуть бути розраховані супротивником заздалегідь. Цей тип схем уразливий до атак компромісу «час-пам'ять», оскільки противник може попередньо обчислити відсутній блок даних до потрібного часу [16].

Таким чином, властивість схеми гешування паролів бути обчислювально складною за пам'яттю є однією з ключових, тому більшість схем, що були запропоновані для гешування паролів протягом декількох минулих років, мають цю властивість.

1.5 Конкурс «Password Hashing Competition»

Через мале розмаїття доступних рішень для гешування паролів, в 2013 році міжнародною криптографічною спільнотою було оголошено конкурс «Password Hashing Competition» (PHC) [1]. Метою цього конкурсу було визначення нових ефективних та безпечних схем гешування паролів, що стануть придатними для широкого використання.

Спочатку було запропоновано 24 безпечні схеми гешування паролів, огляд яких представлений в [13]. Дев'ять з них в наступному раунді у 2014 році були додатково вивчені з точки зору безпеки, простоти, ефективності та додаткових функцій, які вони надають. У 2015 році на основі більш точного аналізу безпеки та оцінки ефективності було оголошено переможця та ще чотири фіналісти, що отримали особливе визнання.

Всі схеми є загальнодоступними без патентних обмежень. Для роботи вони вимагають зазначення принаймні наступних параметрів:

– пароль, зазвичай будь-якого розміру від 0 до 128 байт незалежно від кодування;

– значення солі, зазвичай розміром 16 байт;

– довжина виходу, зазвичай 32 байти;

– один або декілька параметрів для налаштування витрат часу (t_cost) та/або витрат пам'яті (m_cost).

Основні властивості безпеки, якими повинні володіти сучасні схеми гешування паролів, були описані у підрозділі 1.2.

У фінал конкурсу «Password Hashing Competition» вийшли такі схеми: Argon2 [16], Catena [17], Lyra2 [18], Makwa [19] та Yescrypt [20]. За винятком MAKWA, всі інші фіналісти є обчислювально складними за пам'яттю схемами. Переможцем конкурсу була оголошена схема Argon2, інші ж 4 схеми отримали особливе визнання завдяки своїм особливостям дизайну та характеристикам. Конкурс дозволив поглибити знання в сфері гешування паролів та підвищив тенденцію щодо використання обчислювально складних за пам'яттю функцій [10].

До конструкції схеми гешування паролів зазвичай входить певна геш-функція. Розглянемо детально структуру геш-функцію «Купина» з метою її подальшого впровадження у відомі схеми гешування паролів для використання модифікованих схем з метою захисту інформації на національному рівні.

1.6 Геш-функція «Купина» і її структурні особливості

Купина є криптографічною геш-функцією, що визначена національним стандартом ДСТУ 7564:2014 «Інформаційні технології. Криптографічний захист інформації. Функція гешування» [2].

Майже двадцять років Україна у якості основної криптографічної

геш-функції використовувала міждержавний стандарт ГОСТ 34.311-95 (російський ГОСТ Р 34.11-94 [21]). Однак у 2008 році стали відомі теоретичні криптоаналітичні атаки [22], що унеможливило його подальше використання, попри все ж забезпечення практичної стійкості. До того ж ГОСТ 34.311-95 не відповідає сучасним вимогам з точки зору швидкодії реалізації на сучасних програмних платформах загального призначення. На заміну застарілого стандарту розроблена геш-функція «Купина», що була введена у дію у 2015 році, і наразі широко використовується для забезпечення криптографічного захисту на національному рівні.

Купина — це ітеративна геш-функція, заснована на дизайні Меркле-Дамгора. Вона використовує функцію стиснення Девіса-Мейєра, що побудована на основі конструкції блокового шифру Івена-Мансура. Функція стиснення Купини за структурою є підстановлювально-перестановочною мережею, яка відповідає стратегії дизайну AES. Результатом роботи геш-функції є бітова послідовність від 8 до 512 біт. Основними режимами роботи функції гешування, рекомендованими до застосування, є «Купина-256», «Купина-384» і «Купина-512».

Розглянемо основні характеристики конструкції геш-функції «Купина».

На вхід функції гешування подається повідомлення M як бітова послідовність довжини N. Далі повідомлення завжди доповнюється за певними правила до довжини, що кратна розміру блоку, та поділяється на блоки m_1, m_2, …, m_k довжиною l біт кожен, де l визначається відповідно до розміру геш-значення n, $n \in \{8 \cdot s \mid s = 1, 2, \ldots, 64\}$:

$$l = \begin{cases} 512 \text{ для } 8 \leqslant n \leqslant 256, \\ 1024 \text{ для } 256 < n \leqslant 512. \end{cases}$$

Максимальна довжина повідомлення, що може бути оброблено, становить $(2^{96} - 1)$ біт.

Обчислення геш-значення відбувається за такою ітеративною процедурою:

$$
\begin{aligned}
&h_0 = IV,\\
&h_v = T_l^{\oplus}(h_{v-1} \oplus m_v) \oplus T_l^{+}(m_v) \oplus h_{v-1},\ v = 1, 2, \ldots, k,\\
&H(IV,\, M) = R_{l,n}(T_l^{\oplus}(h_k) \oplus h_k),
\end{aligned}
$$

де IV — вектор ініціалізації довжиною l біт, $T_l^{\oplus}$, T_l^{+} — бієктивні перетворення, що виконують відображення вхідного блоку довжиною l біт у вихідний такої самої довжини, $R_{l,n}(x)$ — функція, що повертає n старших біт з вхідного блоку x довжиною l біт ($n < 1$), де результат записується в молодші n біт обчисленого значення. На рисунку 1.2 наведена структурна схема геш-функції «Купина» у загальному вигляді.

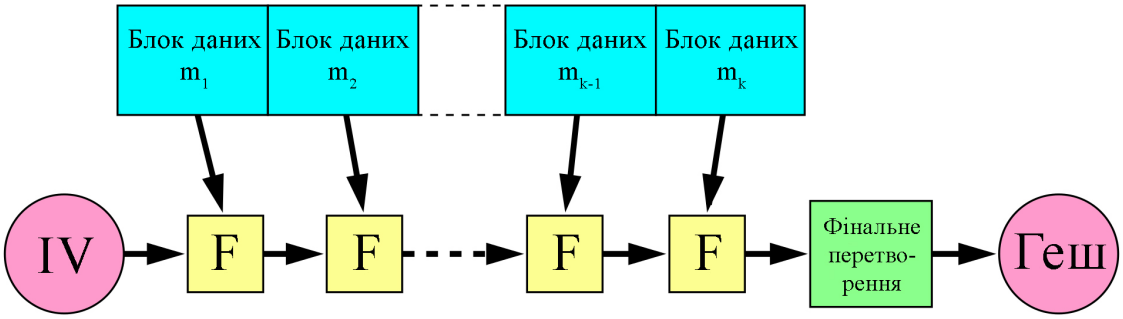

Рисунок 1.2 – Загальна структурна схема геш-функції «Купина» [23]

Як видно, функція стиснення F використовує дві перестановки $T_l^{\oplus}$ та T_l^{+} та обчислюється наступним чином:

$$F(m,\, h) = T_l^{\oplus}(h \oplus m) \oplus T_l^{+}(m) \oplus h.$$

Тоді на k-му кроці $h_k = F(m_k,\, h_{k-1})$, причому $h_0 = IV$. Графічне представлення структури функції стиснення наведено на рисунку 1.3.

Додаткові відомості щодо структури геш-функції можна знайти в [2].

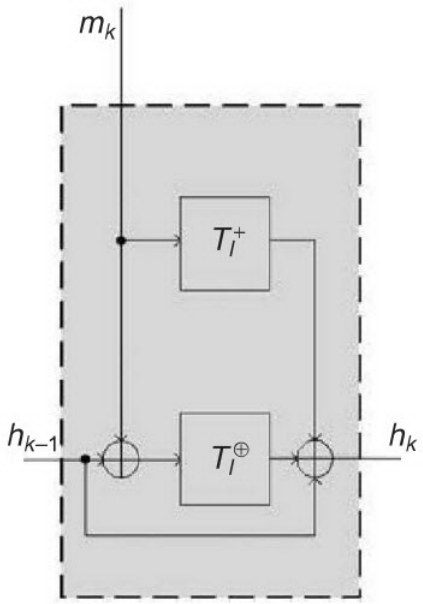

Рисунок 1.3 – Структура функції стиснення геш-функції «Купина» [24]

Висновки до розділу 1

Пароль є секретною фразою користувача, що використовується для його автентифікації та отримання доступу до сервісу. Він вимагає надійного методу обробки та зберігання зі сторони серверу автентифікації для унеможливлення розсекречення та, відповідно, незаконного доступу до облікового запису користувача. Тому для забезпечення безпечного зберігання секретної фрази користувача на сервері її стали обробляти криптографічними геш-функціями та зберігати у подальшому не як відкритий текст, а як геш.

Через відсутність повністю криптографічно безпечної геш-функції, було розроблено цілі схеми для гешування паролів. Однак з часом зловмисники стали ще більше технічно оптимізовані, тому широко розповсюджені схеми виявилися уразливими до різного роду атак, що суттєво збільшило ризики їх подальшого використання та підштовхнуло до розвитку в розробці обчислювально складних за пам'яттю функцій. Згодом було сформульовано бажані властивості безпеки схем гешування паролів, та з 2013 по 2015 роки міжнародною криптографічною

спільнотою було проведено конкурс «Password Hashing Competition» з метою визначення нових ефективних та безпечних схеми, що стануть придатними для широкого використання.

Зазвичай до структури схем гешування паролів входить певна геш-функція. Зокрема, можна розглянути питання впровадження геш-функції «Купина», що описана ДСТУ 7564:2014 «Інформаційні технології. Криптографічний захист інформації. Функція гешування», у схеми гешування паролів, що є фіналістами конкурсу PHC, з метою розробки схеми придатної для захисту інформації в межах нашої країни. Наступний розділ присвячено дослідженню цього питання.

2 ПОРІВНЯЛЬНИЙ АНАЛІЗ ФІНАЛІСТІВ КОНКУРСУ «PASSWORD HASHING COMPETITION»

У даному розділі буде розглянуто структуру кожної зі схем-фіналістів конкурсу «Password Hashing Competition», буде сформульовано критерії для порівняння цих схем, виконано порівняння схем за критеріями та визначено, які зі схем придатні для впровадження геш-функції «Купина». Наприкінці розділу буде наведено алгоритм роботи нових схем.

2.1 Порівняння схем-фіналістів конкурсу PHC

Розглянемо докладно конструкцію та особливості схеми гешуванная паролів — переможця конкурсу — Argon2 [16], а також схем, що отримали особливе визнання — Catena [17], Lyra2 [18], MAKWA [19] та Yescrypt [20].

2.1.1 Критерії порівняння схем

Сформулюємо критерії для порівняння виходячи з вимог, що наразі висуваються до сучасних схем гешування паролів.

Як було зазначено у попередньому розділі, пароль обов'язково повинен оброблятися з додаванням випадкового значення солі для надання певного ступеню унікальності результуючому гешу, а також схема повинна вимагати зазначення витрат часу та/або витрат пам'яті для виконання обчислень для забезпечення стійкості проти добре оптимізованого зловмисника. Тому першим критерієм для порівняння схем є вхідні та вихідні дані, які вони підтримують.

Для розуміння того, яким чином можливо впровадити геш-функцію «Купина» у дані схеми, необхідно визначити, які криптопримітиві та перетворення використовуються схемами. Тому необхідно порівняти схеми за їх складовими елементами.

Також особливу увагу слід звернути на безпеку даних схем, а саме чи мають схеми бажані властивості безпеки, які було описано у підрозділі 1.2. Варто зазначити, що властивість схеми буде крипотографічно безпечною забезпечується використанням у ній криптографічно стійких перетворень, без використання яких розробка схеми не має сенсу. Також варто додати, що при порівнянні схем пункт про їх захист від ЦП-оптимізованих зловмисників буде опущено за причиною, визначеною раніше. Таким чином, ще одним критерієм для порівняння є властивості безпеки схем.

Отже, надалі будемо порівнювати схеми за трьома визначеними вище критеріями.

2.1.2 Відповідність схем парольного гешування сформульованим критеріям порівняння

Спочатку виділимо основні особливості даних схем.

Argon2 є вдосконаленою версією схеми Argon [25], що брала участь у першому етапі конкурсу. У схемі втілені останні досягнення в дизайні обчислювально складних за пам'яттю функції. Argon2 має два варіанти роботи з різним способом доступу до пам'яті — Argon2d та Argon2i. Перший варіант схеми використовує залежний від даних доступ, і він швидший. Другий використовує незалежний від даних доступ до пам'яті, також він повільніший, оскільки виконує більше проходів по пам'яті для захисту від атак компромісу. Для гешування паролів розробниками рекомендовано використовувати варіант Argon2i. Схема Argon2 була

сконструйована з метою максимізації вартості вичерпного пошуку на не x86 архітектурах так, щоб використання спеціалізованих ІССП не давало значних переваг перед здійсненням вичерпного пошуку на машині захисника.

На відміну від Argon2, Catena [17] має простий та елегантний дизайн на основі графів. Існує два варіанти схеми з різним вибором внутрішньої функції, що підтримують незалежний від вхідних даних (паролю) спосіб доступу до пам'яті. Потік даних в варіанті Catena-Butterfly базується на наборі із так званих «double-butterfly»-графів, а в Catena-Dragonfly на так званих «bit-reversal»-графах.

Lyra2 [18] — це схема гешування паролів, що ґрунтується на криптографічних губках. Lyra2 є вдосконаленням варіантом алгоритму Lyra [26], що забезпечує ще більш високий рівень захисту від атак різного роду та подолання деяких обмежень цієї схеми. Конструкція Lyra2 є простою і строго послідовною, унеможливлюючи розпаралелювання для запобігання здійснення паралельних атак. Особливою версією схеми є Lyra2_p, що дає можливість обробки багатоядерних процесів для захисника, збільшуючи витрати атакуючого.

Схема гешування паролів MAKWA [19] вирізняється тим, що її робота базується на піднесенні до квадрату за модулем, що є числом Блюма. Алгоритм працює на великих числах подібних до тих, що використовуються в криптосистемі RSA. Безпечність схеми залежить від засекреченості дільників модуля, оскільки наявність інформації про них пришвидшує роботу зловмисника.

Yescrypt [20] є оптимізованою версією алгоритму Scrypt [9] з покращеним рівнем захисту. Вона успадковує від свого попередника більшу частину структури. За своєю суттю Yescrypt є сімейством функцій, кожна з яких активується комбінацією прапорів.

Таким чином, кожна з цих схем має свою особливу конструкцію та особливості, що відрізняють її від інших, і саме завдяки цьому наведені схеми привернули до себе увагу на конкурсі та отримали особливе

визнання. Представимо основні особливості розглянутих схем у вигляді таблиці 2.1.

Таблиця 2.1 – Основні особливості схем-фіналістів конкурсу PHC

Схема	Ґрунтується на	Варіанти схеми
Argon2	Ітеративна структура обчислення загального виду	Argon2d Argon2i
Catena	Графи	Catena-Butterfly Catena-Dragonfly
Lyra2	Схема губки	Lyra2 Lyra2$_p$
MAKWA	Піднесення до квадрату за модулем	MAKWA
Yescrypt	Алгоритм Scrypt	Yescrypt

Кожна зі схем гешування паролів може обробляти вхідні дані визначеного розміру, а також вимагає зазначення параметрів, необхідних для своєї роботи та виробляє вихідне значення певного розміру. Опишемо цю інформацію для кожної зі схем.

Характеристика вхідних та вихідних даних схеми **Argon2**.

– *Первинні входи:*
  – пароль P довжиною від 0 до $2^{32} - 1$ байт;
  – сіль S довжиною від 8 до $2^{32} - 1$ байт (рекоменд. 16 байт).
– *Параметри:*
  – ступінь паралелізму p — ціле число від 1 до $2^{24} - 1$;
  – довжина тегу τ — ціле число байт від 4 до $2^{32} - 1$;
  – розмір пам'яті m — ціле число кілобайт від $8p$ до $2^{32} - 1$;
  – кількість ітерацій t — ціле число від 1 до $2^{32} - 1$;
  – номер версії v — один байт $0x13$;
  – секретне значення K довжиною від 0 до $2^{32} - 1$ байт;
  – асоційовані дані X довжиною від 0 до $2^{32} - 1$ байт;
  – тип Argon2 y — 0 для Argon2d, 1 для Argon2i.

– *Результат:* тег τ розміром від 4 до $2^{32} - 1$ байт.

Характеристика вхідних та вихідних даних **Catena**.

– *Первинні входи:*
  – пароль pwd довжиною від 0 до 128 байт;
  – сіль s — випадкове значення довжиною 16 байт.
– *Параметри:*
  – вимоги до пам'яті *garlic* g (g_{low}, g_{hight});
  – кількість секретних бітів солі *paper* p;
  – параметр безпеки λ функції F;
  – режим роботи d — дорівнює 0 для гешування паролів;
  – параметр t, що обчислюється певним чином;
  – загальнодоступний вхід γ — наприклад, сіль;
  – довжина m виходу схеми;
  – асоційовані дані AD (необов'язково);
  – секретний ключ K (необов'язково).
– *Результат:* x розміром 32 байти, але не обмежений цим значенням.

Характеристика вхідних та вихідних даних **Lyra2**.

– *Первинні входи:*
  – пароль pwd довільного розміру;
  – сіль $salt$ довільного розміру.
– *Параметри:*
  – розмір блоку губки b — кількість біт;
  – кількість бітів ω для здійснення обертання;
  – витрати часу T — кількості ітерацій, $T > 1$;
  – кількість рядків у матриці пам'яті R — додатне ціле число;
  – кількість стовпців у матриці пам'яті C — додатне ціле число;
  – довжина гешу k — ціле число біт.
– *Результат:* K розміром k байт.

Характеристика вхідних та вихідних даних **MAKWA**.

– *Первинні входи:*
  – пароль π довжиною u в байтах, $u \leq 255$ та $u \leq k - 32$;

– сіль σ довжиною принаймні 16 байт.

– *Параметри:*

– число Блюма p;

– число Блюма q;

– довжина модулю k — ціле число байт від 160 до 256;

– модуль n — добуток чисел p та q довжиною k байт;

– довжина;

– робоча степінь w — невід'ємне ціле число.

– *Результат:* τ розміром до 256 байт.

Характеристика вхідних та вихідних даних **Yescrypt**.

– *Первинні входи:*

– пароль P довільного розміру;

– сіль S довільного розміру.

– *Параметри:*

– витрати ЦП/пам'яті N — ціле число;

– довжина $hLen$ виходу перетворення PRF — число байт;

– довжина $MFLen$ блоку функції MF – число байт;

– довжина $dkLen$ результату — додатне ціле $\leq (2^{32} - 1)hLen$;

– ступінь паралелізму p — додатне ціле $\leq (2^{32} - 1)hLen/MFLen$;

– контроль часу обчислення t — 32-бітове число;

– кількість поточних оновлень, виконаних до гешування, g;

– флаг $flag$ — YESCRYPT_RW або YESCRYPT_WORM.

– *Результат:* DK розміром $dkLen$ байтів.

Метою виділення характеристик даних схем-фіналістів конкурсу PHC було групування параметрів схем гешування паролів задля порівняння схем, а також для зручності при впровадженні схем в ту чи іншу систему.

З наведеної інформації очевидно, що кожна зі схем вимагає зазначення певних необхідних саме для своєї конструкції параметрів. Спільним для цих схем є те, що для них всіх первинними входами є пароль та сіль, а також вони вимагають вказання хоча б одного

параметру для налаштування вимог до часу або простору для здійсненні обчислень.

Надалі розглянемо структуру даних схем та зазначимо основні складові елементи, які входять до конструкцій схем, у таблиці 2.2.

Таблиця 2.2 – Складові елементи схем-фіналістів конкурсу PHC

Схема	Основна геш-функція	Додаткові перетворення
Argon2	Blake2b	Геш-функція H' змінної довжини на основі Blake2b; функція стиснення G, що базується на внутрішній перестановці Blake2b; простий ГПВЧ — функція G у режимі лічильника.
Catena	Blake2b	Геш-функція H' — Blake2b-1; функція Г — SaltMix; функція F — (g,λ)-Double-Butterfly гешування (DBH^g_λ) або (g,λ)-Bit-Reversal гешування (BRH^g_λ); функція $flap$, що включає у себе функції Г та F; функції H_{first} та H_{init}; ГВЧ — функція $xorshift1024star$.
Lyra2	—	Функція губки f — функція стиснення Blake2b з інтеграцією множення у її функцію G — функцію G BlaMka.
MAKWA	SHA-256	HMAC_DRBG з SHA-256; піднесення до квадрату за модулем n.
Yescrypt	SHA-256	алгоритм ROMix; алгоритм SMix з HMAC-SHA256; алгоритм pwxform; алгоритм BlockMix_pwxform з Salsa20/2; алгоритм $PBKDF2_{HMAC-SHA256}$.

З наведеної таблиці видно, що до конструкції майже всіх схем входить певна геш-функція, однак необхідно відмітити наступне:

– у Argon2 геш-функція Blake2b є окремим кроком схеми, приймає участь у здійсненні одного з додаткових перетворень;

– у Catena основна геш-функція, якою є також Blake2b, є як окремим

кроком схеми, так і використовується у складі перетворень функції $flap$;

– у схемі Lyra2 використовується лише модифікована функція стиснення геш-функції Blake2b;

– схема MAKWA використовує геш-функцію SHA-256 лише як частину підрахунку HMAC;

– алгоритм Yescrypt використовує SHA-256 для підрахунку HMAC.

Перейдемо до порівняння схем за їх властивостями безпеки.

Базуючись на вивченні існуючих публікацій щодо аналізу криптографічної безпеки даних схем, зазначимо відповіді на наступні запитання:

– чи є схема обчислювально складною за пам'яттю?

– чи захищена схема від атак компромісу «час-пам'ять»?

– чи захищена схема від атак за побічними каналами?

та порівняємо схеми за наявністю цих властивостей.

Обчислювальна складність за пам'яттю:

– Argon2 є обчислювально складною за пам'яттю, у її розробників було на меті розробити схему саме з такою властивістю;

– в Catena ці властивість забезпечується використанням спеціально визначеної функції F;

– Lyra2 має простий дизайн та у той же час підтримує концепцію обчислюваної складності за пам'яттю;

– схема MAKWA не має даної властивості, оскільки своїми розробниками вона не була сконструйована, щоб бути обчислювально складною за пам'яттю;

– Yescrypt, як було зазначено раніше, є удосконаленим варіантом схеми Scrypt, тому вона наслідувала властивість бути обчислювально складною за пам'яттю.

Захист від атак компромісу «час-пам'ять»:

– схема Argon2 забезпечує стійкість проти таких атак шляхом використання залежного від даних доступу до пам'яті в варіанті схеми Argon2d та виконання декількох проходів по пам'яті в Argon2i;

– у обох варіантах схеми Catena використовується функція SaltMix з випадковим способом доступу до пам'яті, що базується на відкритому вході, що і призводить до ускладнення реалізації компромісу;

– для Lyra2 відомо, що вартість обробки атак з меншою кількістю пам'яті, ніж визначається алгоритмом, зростає набагато швидше, ніж квадратично, перешкоджаючи здійсненню компромісу «час-пам'ять»;

– інформація щодо аналізу захисту MAKWA від атак такого типу відсутня;

– в Yescrypt встановлений прапор YESCRYPT_RW перешкоджає здійсненню атак «час-пам'ять».

Захист від атак за побічними каналами:

– варіант схеми Argon2 Argon2d використання залежного від даних доступу до пам'яті уразливий до таких атак, а Argon2d — ні;

– обидва варіанти схеми Catena використовують незалежний від вхідних даних доступ до пам'яті, тому вони стійкі до даного типу атак;

– пароль та внутрішній стан схеми Lyra2 в процесі роботи перезаписується, що унеможливлює здійснення таких атак;

– стійкість схеми MAKWA проти атак за побічними каналами її розробниками наразі не проаналізована, однак з [13] відомо, що ця схема є стійкою до Garbage-Collector атак (GCA), оскільки початковий або гешований пароль використовується двічі для ініціалізації внутрішнього стану схеми, який потім обробляється шляхом піднесення до квадрату за модулем n.

– Yescrypt забезпечує стійкість проти GCA атак тільки за певних вимог, що відомо з [13].

Представимо відповіді на запитання у вигляді таблиці 2.3.

Будемо використовувати наступні позначення:

– «✓» — схема має дану властивість;

– «✓*» — схема частково має дану властивість;

– «—» — схема не має даної властивості;

– «∗» — немає даних.

Таблиця 2.3 – Властивості безпеки схем-фіналістів конкурсу PHC

Схема		Обчислювально складна за пам'яттю схема	Захист від атак компромісу «час-пам'ять»	Захист від атак за побічними каналами
Argon2	Argon2d	✓	✓	✓
	Argon2i	✓	✓	—
Catena	Catena-B.	✓	✓	✓
	Catena-D.	✓	✓	✓
Lyra2		✓	✓	✓
MAKWA		—	*	✓*
Yescrypt		✓	✓*	✓*

Таким чином, провівши детальний огляд структури схем гешування паролів — фіналістів конкурсу «Password Hashing Competition», для розв'язання задачі впровадження у ці схеми геш-функції «Купина», що визначена ДСТУ 7564:2014 «Інформаційні технології. Криптографічний захист інформації. Функція гешування» [2], найбільш придатними для цього є Argon2 та Catena. Ці схеми добре проаналізовані, мають якісні криптографічні властивості, що відповідають вимогам до сучасних схем гешування паролів, а також, відповідно до рекомендацій розробників, заміна основної геш-функції у цих схемах допускається. Інші схеми гешування паролів не були обрані, оскільки:

– конструкція схеми Lyra2 заснована на новітніх підходах, які ще не достатньо досліджені, тому віддається перевага більш консервативним підходам;

– MAKWA не є обчислювально складною за пам'яттю функцією, що є дуже важливою на даний момент властивістю для схеми гешування паролів;

– в Yescrypt геш-функція є складовою частиною внутрішніх алгоритмів, тому її заміна буде вимагати перепроведення аналізу стійкості всієї схеми.

Перейдемо до впровадження геш-функції «Купина» у обрані схеми.

2.2 Схеми парольного гешування на основі геш-функції «Купина»

В обох схема будемо виконувати заміну геш-функції Blake2b на геш-функцію «Купина». Оскільки Blake2b [27] виробляє геш розміром 64 байти, тому для її заміни доречно обрати геш-функцію «Купина-512». Надалі в тексті під «Купина» будемо розуміти «Купина-512». Основна відмінність цих функцій полягає в структурі раундового перетворення. Раундова функція Blake2b виконує обробку блоку даних, використовуючи такі операції, як додавання за модулем 2^{64}, циклічний зсув праворуч та XOR. У функції гешування «Купина» до вхідних даних застосовуються бієктивні перетворення $T_l^{\oplus}$ та T_l^{+}, кожне з яких реалізоване у вигляді ітеративного застосування низки функцій, що обробляють вхідний аргумент як матрицю визначеного розміру, що містить елементи поля $GF(2^8)$. Більш детально структурні особливості геш-функції «Купина» описано в підрозділі 1.6, а також в [2]. Додаткову інформацію про структуру Blake2b можна отримати з [27].

2.2.1 Модифікація схеми Argon2

Оскільки в схемі гешування паролів Argon2 функція стиснення G базується на внутрішній перестановці геш-функції Blake2b, тому визначимо, чи можна у схемі й її замінити на функцію стиснення F геш-функції «Купина».

Функція G Argon2 працює з двома 1024-байтові блоки та виробляє 1024-байтовий результат. У своїй конструкції вона використовує раундову функцію $\mathcal{P}$ Blake2b. Спочатку обчислюється $R = X \oplus Y$. Потім R представляється як матриця розміру 8×8 із 16-байтових значень R_0, R_1, $\ldots$, R_{63}. Далі застосовується перестановка $\mathcal{P}$ спочатку по рядкам,

а потім по стовпцям, у результаті чого одержується Z. $\mathcal{P}$ у свою чергу представляється як матриця 4×4 64-бітових значень, до елементів $(a,\ b,\ c,\ d)$ якої застосовується функція G', що складається з таких операцій:

– додавання за модулем 2^{64};

– циклічний зсув праворуч;

– урізання до найменших значущих бітів;

– додавання за модулем 2;

– множення за модулем 2^{64}, що є єдиною відмінністю від оригінального дизайну Blake2.

На наступному кроці додаючи Z до R за модулем 2 отримаємо результат роботи функції стиснення. Загалом робота G описується наступним чином:

$$G: \quad (X,Y) \ \rightarrow \ R = X \oplus Y \ \stackrel{\mathcal{P}}{\rightarrow} \ Q \ \stackrel{\mathcal{P}}{\rightarrow} \ Z \ \rightarrow \ Z \oplus R.$$

Детальна інформація щодо обчислення функції G наведена в [16].

На відміну від описаної вище функції G, функцію стиснення F геш-функції «Купина-512» може працювати з двома 1024-бітовими блоками X та Y. В загальному робота функції стиснення F описується наступним чином:

$$F(X,\, Y) = T_l^{\oplus}(Y \oplus X) \oplus T_l^{+}(X) \oplus Y.$$

За схемою Argon2 функція стиснення працює з двома 1024-байтовими блоками, тому внесемо деякі зміни у її структуру функції F.

Нехай F отримує на вхід два 1024-байтові блоки X та Y, як цього вимагає схема Argon2. Далі для здійснення бієктивних перетворень $T_l^{\oplus}$ та T_l^{+} представимо кожен блок даних у вигляді $X = x_1 \parallel x_2 \parallel \cdots \parallel x_8$ та $Y = y_1 \parallel y_2 \parallel \cdots \parallel y_8$. Тоді нехай результат роботи функції визначається як $Z = z_1 \parallel z_2 \parallel \cdots \parallel z_8$, а z_i обчислюється як $z_i = F(x_i \oplus y_{i-1},\ y_i)$, причому $y_0 = y_8$. Виконання обчислення таким способом дозволить

обробляти функції F блоки по 1024-байти, зробить обчислення залежними між собою та не потребуватиме великих обсягів пам'яті для зберігання даних.

На рисунку 2.1 наведемо графічне представлення принципу роботи описаних функцій.

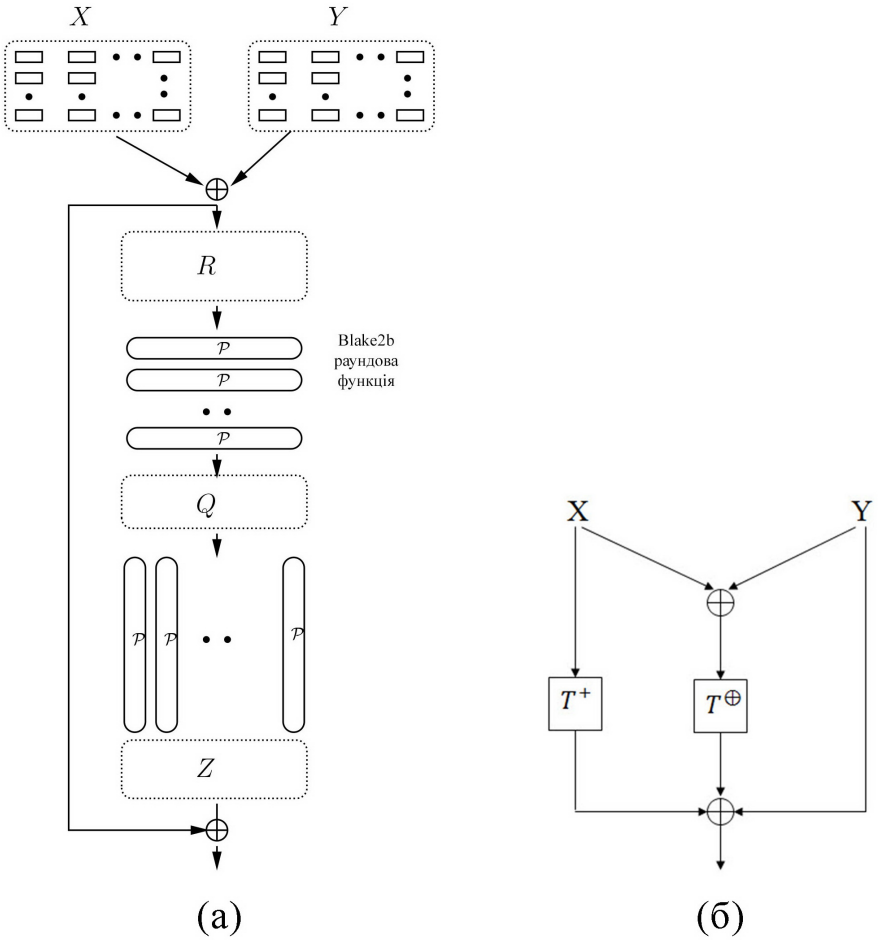

Рисунок 2.1 – Функції стиснення, що використовуються в: (а) схемі Argon2 [28], (б) геш-функції «Купина»

Модифікована під роботу схеми Argon2 функція F має хороші криптографічні властивості, оскільки у своїй конструкції вона має лінійні та нелінійні перетворення. Однак у порівнянні з G, функція F є повільнішою, оскільки перетворення $T_l^{\oplus}$ та T_l^{+} вимагають значного часу для здійснення через роботу з елементами поля $GF(2^8)$.

В специфікації Argon2 розробниками зазначено, що вони

дозволяють вибрати іншу функцію стиснення G та геш-функцію H. Отже, в схемі Argon2 замінимо геш-функцію Blake2b та функцію стиснення G на геш-функцію «Купина» та її функцію стиснення F відповідно. Зробивши ці зміни, отримаємо нову схему, яку через її структурні особливості назвемо ArgoKupyna.

В загальному процес роботи нової схеми складається з наступних кроків:

1) ініціалізація вхідних параметрів;

2) обробка вхідних даних геш-функцією «Купина»;

3) заповнення пам'яті з використанням функції стиснення F геш-функції «Купина»;

4) підрахунок вихідного значення, тегу τ, з використанням геш-функції «Купина».

Розглянемо кожен крок схеми докладніше.

На першому кроці для роботи схеми необхідно зазначити вхідні дані, що було перелічено у підрозділі 2.1.2.

На другому кроці виконується обробка вхідних даних разом з їх довжинами за допомогою геш-функції «Купина», процедуру підрахунку результату якої було наведено у підрозділі 1.6. У результаті отримуємо значення H_0.

На третьому кроці здійснюється заповнення пам'яті, що складається з $m' = \lfloor \frac{m}{4p} \rfloor \cdot 4p$ 1024-байтових блоків та організована як матриця $M[i][j]$ з p рядків та $q = m'/p$ стовпців. Блок, який виробляється при проході t, позначається як $M^t[i][j]$, $t > 0$. Блоки обчислюються наступним чином:

$$M^1[i][0] = H'(H_0 || \underbrace{0}_{\text{4 байти}} || \underbrace{i}_{\text{4 байти}}), \quad 0 \le i < p;$$

$$M^1[i][1] = H'(H_0 || \underbrace{1}_{\text{4 байти}} || \underbrace{i}_{\text{4 байти}}), \quad 0 \le i < p;$$

$$M^1[i][j] = F(M^1[i][j-1], M^1[i'][j']), \quad 0 \le i < p,\ 2 \le j < q,$$

де індекси блоку $[i'][j']$ визначаються по-різному для Argon2d та Argon2i, F

— функція стиснення геш-функції «Купина», що була описана вище, а H' — геш-функція змінної довжини, яка будується визначеним чином на основі зазначеної геш-функції. Оскільки було обрано геш-функцію «Купина-512», тобто максимальний розмір виходу, що підтримується складає 64 байти, що відповідає Blake2b, то визначення геш-функції H' в ArgoKupyna відповідає Argon2.

Якщо $t > 1$, процедура повторюється, але новий блок додається за модулем 2 до старого, а не перезаписується:

$$M^t[i][0] = F(M^{t-1}[i][q-1], M[i'][j']) \oplus M^{t-1}[i][0];$$
$$M^t[i][j] = F(M^t[i][j-1], M[i'][j']) \oplus M^{t-1}[i][j].$$

Тут блок $M[i'][j']$ може бути або $M^t[i'][j']$ для $j' < j$ або $M^{t-1}[i'][j']$ для $j > j'$.

Щоб зробити можливим паралельне обчислення блоків, матриця пам'яті розділяється на долі $S = 4$. На перетині долі та смуги розташовується сегмент довжини q/S. Сегменти одної долі обчислюються паралельно, і вони не можуть посилатися один на одного, але можуть посилатися на сегменти з інших долей. Обчислення індексів відбувається наступним чином. У Argon2d обираються перші 32 біти блоку $M[i][j-1]$ і позначається це значення як J_1. Наступні 32 біти $M[i][j-1]$ позначаються як J_2. У Argon2i запускається F^2 — два раунди функції стиснення F — у режимі лічильника, де перший вхід повністю складається з нулів, а другий визначається певним чином. Кожен запуск F^2 дає 128 64-бітових значень $J_1||J_2$. Далі за визначеними правилами відбувається відображення J_1 та J_2 в індекс блоку для посилання.

На четвертому кроці застосуванням H' до M_{final}, що є результатом додавання за модулем 2 елементів матриці з останнього стовпця після виконання T ітерацій по заповненню пам'яті, отримуємо Тег τ, що є результатом роботи схеми:

$$H'(M_{\text{final}}) \to \text{Тег } \tau.$$

Графічне представлення процесу роботи ArgoKupyna відповідає Argon2 та наведено на рисунку 2.2.

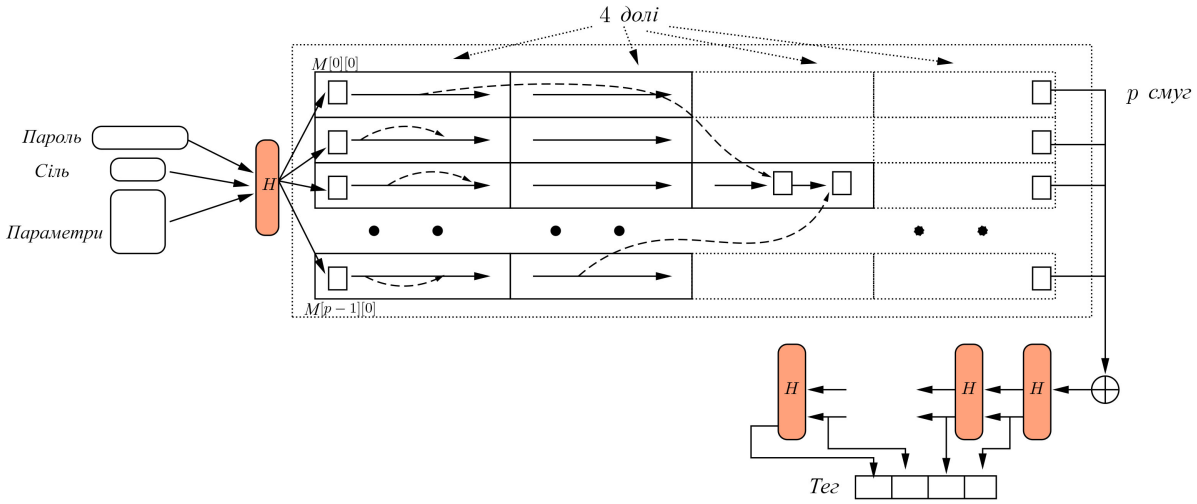

Рисунок 2.2 – Один прохід схеми Argon2 з p смугами та 4 долями [29]

Додаткові відомості щодо виконання обчислень за схемою Argon2 наведено в [16].

Перейдемо до впровадження геш-функції «Купина» у схему гешування паролів Catena.

2.2.2 Модифікація схеми Catena

В специфікації схеми Catena розробниками зазначено, що вони дозволяють змінювати H, H' на будь-які підходящі. Отже, в схемі Catena будемо використовувати геш-функцію «Купина» замість Blake2b. Зробивши такі зміни, отримаємо нову схему, яку через її структурні особливості назвемо CatenaKupyna.

Процес роботи нової схеми складається описується наступними кроками:

1) ініціалізація вхідних параметрів;

2) обробка вхідних даних геш-функцією «Купина»;

3) обробка даних функцією $flap$;

4) підрахунок вихідного значення шляхом обробка даних послідовно функцією $flap$, геш-функцією «Купина» та функцією $truncate$ в циклі.

Розглянемо докладніше кожен крок схеми.

Спочатку треба зазначити всі параметри, необхідні для роботи. Параметр t є мультибайтовим значенням, що визначається наступним чином:

$$t \leftarrow H(V) \parallel d \parallel \lambda \parallel m \parallel |s| \parallel H(AD),$$

де $H(V)$ — n-бітове геш-значення ідентифікатора версії, причому для Catena-Dragonfly V визначається, як "Dragonfly", а для Catena-Butterfly — "Butterfly".

На наступному кроці, використовуючи геш-функцію «Купина», обчислюється значення x як:

$$x \leftarrow H(t \parallel pwd \parallel s).$$

Процедуру підрахунку результату геш-функцією було описано у підрозділі 1.6.

На третьому кроці роботи схеми отримане раніше значення x разом з двома іншими параметрами поступає на вхід функції $flap$, у результаті чого x оновлюється, тобто:

$$x \leftarrow flap(\lceil g_{low}/2 \rceil, x, \gamma)$$

На останньому кроці схеми організується цикл по g, від g_{low} до g_{high}, де отримане на минулому кроці значення x оновлюється послідовно за допомогою функцій $flap$, «Купина» та $truncate$ з певними вхідними

параметрами наступним чином:

$$x \leftarrow flap(g, x \parallel 0^*, \gamma),$$
$$x \leftarrow H(g \parallel x),$$
$$x \leftarrow truncate(x, m),$$

де в функції $flap$ x доповнюється нулями так, щоб $x \parallel 0^*$ відповідало довжині виходу геш-функції, а функція $truncate(x, m)$ виводить m найменших значущих байтів x.

У свою чергу функція $flap$, що отримує на вхід g, x та γ, обчислюється наступним чином:

$$(v_{-2}, v_{-1}) \leftarrow H_{init}(x);$$
$$\text{для } i = 0, \ \ldots, 2^g - 1 : \ v_i \leftarrow H'(v_{i-1} \parallel v_{i-2});$$
$$v \leftarrow \Gamma(g, v, \gamma);$$
$$x \leftarrow F(v).$$

Геш-функція H_{init}, що використовується у здійсненні обчислень функцією $flap$, перетворює вхідне значення x довжиною n-біт на два k-бітових значення v_{-2}, v_{-1} наступним чином:

$$l = 2 \cdot k/n;$$
$$\text{для } i = 0, \ \ldots, l - 1 : \ w_i \leftarrow H(i \parallel x);$$
$$v_{-1} \leftarrow (w_0, \ \ldots, w_{l/2-1});$$
$$v_{-2} \leftarrow (w_{l/2}, \ \ldots, w_{l-1}),$$

де k є довжиною виходу H' в бітах.

Як було зазначено раніше, в схемі Catena у якості H використовується геш-функція Blake2b, а у якості H' визначено Blake2b-1, що складається з одного раунду Blake2b, включаючи фінальне перетворення. Більш детальна інформація щодо структури цих функції наведена в [17]. Таким чином, з наведеного випливає, що в схемі

гешування CatenaKupyna ніщо не заперечує у якості H' використовувати подібно скорочену версію геш-функції «Купина».

В алгоритмі $flap$ схеми Catena функція F визначається відповідно до версії схеми. Так, для Catena-Dragonfly F є BRH^g_λ, а для Catena-Butterfly — DBH^g_λ. Обидві функції для свого обчислення використовують геш-функції H_{first} та H'. Функція H_{first} приймає в якості вхідних даних два k-бітових слова v_α і v_β і обчислює r_0 довжиною k-біт, що складається з l псевдовипадкових різних між собою значень $w_0, \ldots, w_{l-1}$. Функція H_{first} у своїх обчисленнях використовує основну геш-функцію Balke2b, яку заміняємо на геш-функцію «Купина».

Функція ж Г для обох варіантів схеми Catena визначається однаково. Перенесемо це і на схему CatenaKupyna. Метою застосування функції Г є оновлення стану масиву v, отримуючи доступ до його елементів залежним від солі чином. В алгоритмі роботи схеми Г представляється функцією $SaltMix$, що для свого обчислення використовує геш-функції H, H' та функцію $xorshift1024star(r, p)$, яка виступає у якості генератора випадкових чисел.

Додаткові відомості щодо обчислення функцій F та Г наведено в [17].

Таким чином, заміна основної геш-функції Blake2b схеми Catena на геш-функцію «Купина» не змінює основну ідею дизайну схеми з використанням функції $flap$, що наведена на рисунку 2.3.

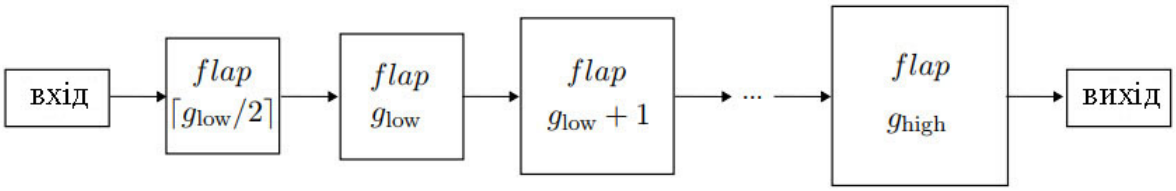

Рисунок 2.3 – Загальний вигляд дизайну схеми гешування паролів Catena, що використовує функцію $flap$ [30]

Висновки до розділу 2

Виконавши огляд структури схем-фіналістів конкурсу «Password Hashing Competition» та порівнявши схеми за визначеними критеріями було визначено, що найбільш придатними для впровадження геш-функції «Купина» є схеми Argon2 і Catena, оскільки вони мають якісні криптографічні властивості та геш-функція у них є окремим заміняймим модулем, тому її заміну буде можливо легко виконати зі збереженням існуючих властивостей безпеки схем.

В обох схемах було виконано заміну основної геш-функції Blake2b на геш-функцію «Купина-512», однак в Argon2 також було замінено функцію стиснення G на геш-функцію «Купина-512» та її модифіковану для роботи зі 1024-байтовими блоками функцію стиснення F.

У результаті було отримано схеми гешування паролів ArgoKupyna та CatenaKupyna та наприкінці розділу сформульовано алгоритм роботи кожної з отриманих схем.

Обидві нові схеми зберігають криптографічні властивості початкових схем, оскільки було замінено один криптографічний примітив з якісними криптографічними властивостями на такий же самий. Однак відмінність в них полягає у швидкості роботи, оскільки бієктивні перетворення геш-функції «Купина» потребують значного часу для виконання, тому ця функція не може забезпечити подібну Blake2b швидкість роботи, але це зовсім не впливає на криптографічні властивості схеми парольного гешування.

ВИСНОВКИ

У ході даної роботи було проведено аналіз опублікованих джерел за темою гешування паролів. Через відсутність повністю безпечної геш-функції пропонується використовувати спеціальні схеми для гешування паролів. Визначено, що найбільше використання отримали такі схеми як PBKDF2, Scrypt та Bcrypt. Однак з часом вони виявилися уразливими до різного роду атак, що суттєво збільшило ризики їх подальшого використання. Тому було проведено ознайомлення з бажаними властивостями безпеки схем гешування паролів, що сформульовані відповідно до атак, існуючих та потенційних. Як виявилося, через мале розмаїття доступних схем у 2013 році міжнародною криптографічною спільнотою було оголошено конкурс «Password Hashing Competition». Метою конкурсу було визначення нових ефективних та безпечних схеми, що стануть придатними для широкого використання. У процесі роботи було розглянуто конструкції схем-фіналістів цього конкурсу. Визначено, що до їх структури входить певна геш-функція, тому було проведено порівняння цих схем за можливістю впровадження у них геш-функції «Купина», що описана у ДСТУ 7564:2014 «Інформаційні технології. Криптографічний захист інформації. Функція гешування» задля розробки нових схем, що стануть придатними для захисту інформації на державному рівні. Було визначено, що такими схемами є Argon2 та Catena. Замінивши в них деякі перетворення, отримано нові схеми ArgoKupyna та CatenaKupyna зі збереженими криптографічними властивостями схем-попередників та описано їх алгоритм роботи.

Результати даної роботи можуть використовуватись для гешування паролів у засобах криптографічного захисту, призначених для державних органів України.

Подальші дослідження можуть бути спрямовані на розробку нових схем гешування паролів та методів їх аналізу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Password Hashing Competition [Electron. resource]: website. — Access link: https://password-hashing.net/.

2. ДСТУ 7564:2014. Інформаційні технології. Криптографічний захист інформації. Функція гешування. — Введ. 01–04–2015. — К. : Мінекономрозвитку України, 2015.

3. Kelsey J., Schneier B., Hall C., Wagner D. Secure Applications of Low-Entropy Keys [Electron. resource]: report / Information Security Workshop (ISW'97). — 1997. — Access link: https://www.schneier.com/academic/paperfiles/paper-low-entropy.pdf.

4. Hellman M. E. A cryptanalytic time-memory trade-off [Electron. resource]: report. / IEEE Transactions on Information Theory. — vol. 26, no. 4. — 1980. — Access link: https://ee.stanford.edu/~hellman/publications/36.pdf.

5. Oechslin P. Making a faster cryptanalytic time-memory trade-off [Electron. resource]: report. / Advances in Cryptology — CRYPTO 2003. — 2003. — Access link: https://lasec.epfl.ch/~oechslin/publications/crypto03.pdf.

6. Схема обробки паролів з використання солі [Електронний ресурс]: Hash-plus-Salt.png / Wordfence. — 1340 x 772. — 24 КБ. — Режим доступу: https://www.wordfence.com/wp-content/uploads/2015/12/Hash-plus-Salt.png.

7. Kaliski B. RFC 2898 — PKCS #5: Password-Based Cryptography Specification Version 2.0 [Electron. resource]: technical report / RSA Laboratories. — 2000. — Access link: https://tools.ietf.org/html/rfc2898.

8. Provos N., Mazieres D. A Future-Adaptable Password Scheme [Electron. resource]: report / In Proceedings of the USENIX Annual Technical Conference. — 1999. — Access link: https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf.

9. Percival C. Stronger key derivation via sequential memory-hard

functions [Electron. resource]: report. — 2009. — Access link: https://www.tarsnap.com/scrypt/scrypt.pdf.

10. Hatzivasilis G. Password-Hashing Status [Electron. resource]: report / Cryptography MDPI. — 2017. — Access link: http://www.mdpi.com/2410-387X/1/2/10/pdf.

11. Wetzels J. Open Sesame: The Password Hashing Competition and Argon2 [Electron. resource]: report / Cryptology ePrint Archive. — 2016. — Access link: https://eprint.iacr.org/2016/104.pdf.

12. Bernstein D.J. Cache-timing attacks on AES [Electron. resource]: report. — 2005. — Access link: http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

13. Forler C., List E., Lucks S., Wenzel J. Overview of the Candidates for the Password Hashing Competition And Their Resistance Against Garbage-Collector Attacks [Electron. resource]: report / Cryptology ePrint Archive. — 2014. — Access link: https://eprint.iacr.org/2014/881.pdf.

14. Denning D.E. Cryptography and Data Security [Electron. resource]: book / Addison-Wesley. — 1982. — Access link: http://faculty.nps.edu/dedennin/publications/Denning-CryptographyDataSecurity.pdf.

15. Chang D., Jati A., Mishra S., Sanadhya S.K. Cryptanalytic Time-Memory Tradeoff for Password Hashing Schemes [Electron. resource]: report / Cryptology ePrint Archive. — 2017. — Access link: https://eprint.iacr.org/2017/603.pdf.

16. Biryukov A., Dinu D., Khovratovich D. Argon2: the memory-hard function for password hashing and other applications [Electron. resource]: report / University of Luxembourg. — 2017. — Access link: https://www.cryptolux.org/images/0/0d/Argon2.pdf.

17. Forler C., Lucks S., Wenzel J. The Catena Password-Scrambling Framework [Electron. resource]: report / Bauhaus-Universität Weimar. — 2017. — Access link: https://www.uni-weimar.de/fileadmin/user/fak/medien/professuren/Mediensicherheit/Research/Publications/catena-v3.3.pdf.

18. Simplicio M.A., Almeida L.C., Andrade E.R., Santos P.C.F., Barreto

P.S.L.M. Lyra2: Password Hashing Scheme with improved security against time-memory trade-offs [Electron. resource]: report / Cryptology ePrint Archive. — 2015. — Access link: https://eprint.iacr.org/2015/136.pdf.

19. Pornin T. The MAKWA Password Hashing Function [Electron. resource]: report. — 2015. — Access link: http://www.bolet.org/makwa/makwa-spec-20150422.pdf.

20. Peslyak, A. Yescrypt — a Password Hashing Competition submission [Electron. resource]: report. — 2015. — Access link: https://password-hashing.net/submissions/specs/yescrypt-v2.pdf.

21. ГОСТ Р 34.11–94. Информационная технология. Криптографическая защита информации. Функция хэширования. — Введ. 01–01–1995. — М., 1994. — 20 с.

22. Mendel F., Pramstaller N., Rechberger Ch., et.al. Cryptanalysis of GOST hash function [Electron. resource]: report /Advances in Cryptology — CRYPTO 2008. — 2008. — Access link: https://iacr.org/archive/crypto2008/51570163/51570163.pdf.

23. Геш будова Меркла-Демґарда [Електронний ресурс]: Merkle-Damgard_hash_big.svg / Davidgothberg. — 8 квітня 2007 року. — 2000 x 633. — 61 КБ. — Режим доступу: https://upload.wikimedia.org/wikipedia/commons/e/ed/Merkle-Damgard_hash_big.svg.

24. ДСТУ 7564:2014. Інформаційні технології. Криптографічний захист інформації. Функція гешування. — Введ. 01–04–2015. — К. : Мінекономрозвитку України, 2015. — С.4.

25. Biryukov A., Khovratovich D. Argon v1: Password Hashing Scheme [Electron. resource]: report / University of Luxembourg. — 2014. — Access link: https://www.cryptolux.org/images/0/0c/Argon-v1.pdf.

26. Almeida, L., Andrade, E., Barreto, P., Simplicio M. Lyra: Password-Based Key Derivation with Tunable Memory and Processing Costs [Electron. resource]: report / Journal of Cryptographic Engineering. — 2014. — Access link: https://eprint.iacr.org/2014/030.pdf.

27. Aumasson J., Neves S., Wilcox-O'Hearn Z., Winnerlein Ch. BLAKE2:

simpler, smaller, fast as MD5 [Electron. resource]: report. — 2013. — Access link: https://blake2.net/blake2.pdf.

28. Biryukov A., Dinu D., Khovratovich D. Argon2: the memory-hard function for password hashing and other applications [Electron. resource]: report / University of Luxembourg. — 2017. — Access link: https://www.cryptolux.org/images/0/0d/Argon2.pdf. — P. 8.

29. Biryukov A., Dinu D., Khovratovich D. Argon2: the memory-hard function for password hashing and other applications [Electron. resource]: report / University of Luxembourg. — 2017. — Access link: https://www.cryptolux.org/images/0/0d/Argon2.pdf. — P. 6.

30. Forler C., Lucks S., Wenzel J. The Catena Password-Scrambling Framework [Electron. resource]: report / Bauhaus-Universität Weimar. — 2017. — Access link: https://www.uni-weimar.de/fileadmin/user/fak/medien/professuren/Mediensicherheit/Research/Publications/catena-v3.3.pdf. — P. 25.